**Лабораторная работа №4**

**КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ НЕЙРОСЕТЕЙ**

*Принципы разработки программного обеспечения, выполняющего имитационное моделирование нейросетей. Структура и функции блоков программы. Пример программной реализации алгоритма обучения персептрона.*

Значительная доля всех приложений нейронных сетей приходится на использование их программных моделей, обычно называемых *нейроимитаторами*. Разработка программы обычно стоит дешевле, а получаемый продукт представляется более наглядным, мобильным и удобным, нежели специализированная аппаратура. В любом случае, разработке аппаратной реализации нейросети всегда должна предшествовать ее всесторонняя отработка на основе теории с использованием компьютерной модели.

Здесь описываются наиболее общие принципы разработки относительно небольших нейропрограмм, обычно индивидуального использования. С целью максимального упрощения изложения выбрана простая архитектура нейронной сети - однослойный ПЕРСЕПТРОН. Теоретические основы этой сети были рассмотрены на лекции.

### Принципы разработки нейроимитаторов.

Нейроимитатор представляет собой компьютерную программу (или пакет программ), которая выполняет следующие функции:

* Описание и формирование архитектуры нейронной сети
* Сбор данных для обучающей выборки
* Обучение выбранной нейросети на обучающей выборке или загрузка уже обученной сети с диска
* Тестирование обученной нейросети
* Визуализация процесса обучения и тестирования
* Решение задач обученной сетью
* Запись результатов обучения и полученных решений на диске.

Промышленные нейроимитаторы (такие, как Neural Works Professional II+ фирмы Neural Ware, или MultiNeuron, разработанный в Красноярском научном центре) предоставляют исследователю широкий набор возможностей по реализации этих функций. В индивидуальных программах, когда пользователя интересует прежде всего результат работы нейросети, часть этих функций может быть максимально упрощена.

Решение задачи с примененнием нейронной сети может состоять из следующих этапов (не обязательно всех и не обязательно выполняемых в указанном порядке).

### Постановка задачи в терминах нейронной сети.

Постановка задачи для нейронной сети имеет определенную специфику, как читатель уже мог убедиться на протяжении всего курса. Прежде всего, необходимо решить, относится ли решаемая задача к одному из стандартных типов нейросетевых постановок: задачи классификации (категоризации), задачи построения функциональной модели (идентификации систем), задачи прогноза, задачи оптимизации и нейроматематики, задачи управления и, наконец, задачи распознавания образов и обработки сигналов.

Нестандартная постановка задачи для нейроЭВМ обычно требует проведения специальных исследований и большого опыта решения других задач. На этом этапе обязательно нужно ответить на вопрос: а нужна ли вообще для решения данной задачи нейронная сеть? Вполне возможно (и часто бывает так), что решение может быть получено алгоритмическим способом. В этом случае применение нейроимитатора обычно оказывается не эффективным.

Далее следует определить используемые в задаче признаковые пространства, в которые включаются параметры, играющие важную роль в данной задаче. Если вы не являетесь экспертом в рассматриваемой предметной области, то на этом этапе целесообразно получить консультации. Общение с коллегами никогда не будет вредным, даже если вас считают ведущим специалистом по данному вопросу.

При построении признаковых пространств следует учесть наличие и доступность соотвествующих данных, в противном случае у вас не будет информации для обучения нейросети.

И, наконец, очень полезно представить ожидаемый результат работы нейросети и способ его дальнейшего использования. Во многих случаях это приводит к упрощению постановки, и, как следствие, к более эффективному решению. Если же полученные результаты не будут соответствовать вашим ожиданиям, то это - важная причина более фундаментально подойти к задаче.

### Выбор и анализ нейроархитектуры, адекватной задаче.

Тип используемой нейросети во много диктуется поставленой задачей. Так, для задачи классификации удобными могут оказаться многослойный персептрон и сеть Липпмана-Хемминга. Персептрон таже применим и для задач идентификации систем и прогноза. При решении задач категоризации потребуются карта Кохонена, архитектура встречного распространения или сеть с адаптивным резонансом. Задачи нейроматематики обычно решаются с использованием различных модификаций модели Хопфилда.

Лучше использовать те архитектуры, свойства котоых вам наиболее знакомы, так как это упростит интерпретацию результатов. На выбо может повлиять наличие или отсутсвие в вашем распоряжении соотвествующих программ.

### Отбор данных и формирование обучающей выборки.

Идеальной является ситуация, когда вы можете получить произвольно много различных данных для вашей задачи. В этом случае следует позаботиться об отсутствии систематических ошибок и уклонений в данных (если только именно этот вопрос не является предметом ваших исследований). Целесообразно включение в обучающую выборку прежде всего тех данных, которые описывают условия, близкие к условиям дальнейшего использования нейросистемы.

Для решения некоторых задач распознавания образов данные, если это возможно, следует представить в инвариантном виде.

Для практических целей следует часть обучающей выборки не использовать при обучении, а применить для последующего тестирования работы нейросети. Полезно понимать, что очень большая выборка обучающих данных сильно замедлит процесс обучения без существенного улучшения результата.

Если в вашем распоряжении имеется весьма ограниченный объем данных, то потребуется анализ его достаточности для решения вашей задачи. Обычно это оказывается весьма непростым вопросом. Одним из решений может быть уменьшение размерности признаковых пространств задачи. В любом случае, обучающих данных должно быть больше, чем обучаемых параметров нейросети.

### Разработка собственной программы или использование существующего нейроимитатора?

Имеющиеся на рынке нейроимитаторы разрабатываются профессионалами специально для вашего удобства. Для практических целей лучше предпочесть их использование. Это обеспечит выполнение стандартов и доказательность полученных вами результатов.

Исключение составляют нестандартные задачи и специализированные архитектуры нейросетей, в этом случае необходимо разрабатывать новую программу. При выборе технической среды для вашего проекта полезно учитывать имеющиеся инструментальные средства для написания нейропрограмм и обработки баз данных. В качестве языка программирования чаще всего используется C (или C++). Для небольших проектов можно выбрать ПАСКАЛЬ или БЕЙСИК.

И, пожалуйста, не тратьте время на перепрограммирование стандартной функции вычисления квадратного корня!

### Анализ результатов.

Это одна из самых важных фаз решения задачи. Для полноты анализа следует позаботиться о нагладности результатов, используя представление их в графическом виде. Если результаты будут использоваться в дальнейших вычислениях с примененим ЭВМ, целесообразно сразу представить их в формате, понимаемом другими программами. Для обмена между программами небольшими таблицами данных можно использовать текстовое представление. Для больших объемов лучше применить стандартные форматы, например, формат dbf-файлов системы dBASE разработки фирмы Ashton-Tate. Это автоматически позволит использовать вам средства этой (и многих других) системы для представления, хранения и редактирования данных.

Если полученные результаты существенно отличаются от ожидаемых, скорее всего придется вернуться к постановке задачи.

Однако возможно, что вы на пороге нового открытия...

### Описание программы PERC.

В этом пункте будет описана простейшая программа PERC, реализующая обучение однослойного ПЕРСЕПТРОНА. В качестве примера была выбрана следующая задача. Нейронной сети предъявляется вектор, состоящий из 10 копмонент, каждая из которых может быть нулем или единицей. Сеть должна научиться определять, чего больше - нулей или единиц.

Для решения такой задачи необходим по крайней мере один нейрон с десятью входами и одним выходом (хотя программа позволяет использовать несколько нейронов). Представляемая функция относится к классу линейно разделимых, поэтому этого одного нейрона достаточно для решения.

В качестве обучающей выборки используются 200 векторов, компоненты которых разыгрываются с использованием датчика псевдослучайных чисел ПАСКАЛЯ. Правильный ответ определяется непосредственным сравнением числа нулей и единиц.

Обучение сети проводится по дельта-правилу Ф.Розенблатта, подробно рассмотренному в лекции 4. По завершении обучения программа выдает число выполненных итераций и значение достигнутой ошибки обучения. В конце этого пункта приведен полный листинг программы PERC и результаты ее работы (Внимание! Если вы проведете расчет по программе на вашем компьютере, то полученные значения могут слегка отличаться от приведенных из-за различных последовательностей случайных чисел).

Для тестирования качества обучения разработана отдельная программа TEST (текст и результаты работы которой тоже приводятся). Структуры используемых данных и работа программы аналогичны программе PERC. Для тестирования также используются случайные вектора.

Результаты теста весьма удовлетворительны, нейронная сеть успешно справляется с задачей с точностью до ошибок во 2-3 знаке ответа. Интерпритация этих ошибок не вызывает затруднений или недоразумений.

#### Текст программы PERC.

PROGRAM PERC;

(\* P E R C - Учебная программа, реализующая однослойный

PERCEPTRON.

ДАТА: 26 октября 1994 г.

АВТОР: С.А.Терехов (email: sta@ch70.chel.su)

\*)

CONST

CMaxInp = 20; (\* Максимальное число входов \*)

CMaxOut = 10; (\* Максимальное число выходов \*)

CMaxImages = 200; (\* Максимальное число образов \*)

CEta = 0.75; (\* Темп обучения \*)

CError = 5.0e-3; (\* Граница требуемой ошибки \*)

CCounter = 1000; (\* Максимальное число итераций \*)

CInitWeight = 5.0; (\* Максимальное начальное значение

случайных синаптических весов \*)

CBiasNeuron = 1.0; (\* Активность нейрона-порога \*)

TYPE

TMatrix = ARRAY[0..CMaxInp,1..CMaxOut] OF REAL;

(\* Нулевой столбец содержит значения порогов \*)

TInpVector = ARRAY[1..CMaxInp] OF REAL;

TOutVector = ARRAY[1..CMaxOut] OF REAL;

(\* Структура сети \*)

TPerceptron = RECORD

NInp : INTEGER; (\* Число входов \*)

NOut : INTEGER; (\* Число выходов \*)

Inp : TInpVector; (\* Текущий вектор входов \*)

Out : TOutVector; (\* Текущий вектор выходов \*)

W : Tmatrix; (\* Матрица связей \*)

END;

(\* Запись в базе данных - обучающей выборке \*)

TBaseRecord = RECORD

X : TInpVector;

Y : TOutVector;

END;

(\* Структура базы данных \*)

TBase = RECORD

NImages : INTEGER; (\* Число обучающих образов \*)

Images : ARRAY[1..CMaxImages] OF TBaseRecord;

END;

VAR

VNet : TPerceptron;

VBase : TBase;

VOK : BOOLEAN;

VError, VTemp, VDelta : REAL;

VCounter, Vi, Vj, Vk : INTEGER;

VFile : FILE OF TPerceptron;

PROCEDURE InitAll;

(\* Инициализация нейронной сети с 10 входами и одним выходом,

задание начальных случайных значений матрицы связей \*)

VAR

Li, Lj, Lk : INTEGER;

BEGIN

WITH VNet, VBase DO

BEGIN

NInp := 10;

NOut := 1;

FOR Li := 0 TO NInp DO

FOR Lj := 1 TO NOut DO

W[Li,Lj] := CInitWeight\*(RANDOM-0.5);

END;

VOK := TRUE;

END;

PROCEDURE GetDataBase;

(\* Генерация обучающей выборки из 200 случайных образов.

При определении правильного числа единиц используется прямой

подстчет \*)

VAR

Li, Lj, Lk : INTEGER;

BEGIN

VOK := TRUE;

WITH VBase, VNet DO

BEGIN

NImages := 200;

FOR Li:= 1 TO NImages DO

BEGIN

Lk := 0;

FOR Lj:=1 TO NInp DO

BEGIN

(\* Случайно 0 или 1 \*)

Images[Li].X[Lj] := RANDOM( 2 );

(\* Подсчет единиц \*)

IF ( Images[Li].X[Lj] > 0 )

THEN Lk := Lk + 1;

END;

(\* Выход равен единице, если в данном входном векторе

число единиц больше числа нулей \*)

IF ( Lk > (NInp-Lk) )

THEN Images[Li].Y[1] := 1

ELSE Images[Li].Y[1] := 0

END;

END;

END;

PROCEDURE SaveNet;

(\* Запись параметров нейронной сети в файл SAMPLE.DAT.

Производится контроль за операциями вывода с использованием

ключа I+ и I- компилятора ТУРБО ПАСКАЛЯ \*)

BEGIN

ASSIGN( VFile, 'SAMPLE.DAT' );

{$I-}

REWRITE( VFile );

{$I+}

VOK := (IOResult = 0);

IF VOK THEN

BEGIN

{$I-}

WRITE( VFile, VNet );

CLOSE ( VFile );

{$I+}

VOK := (IOResult = 0);

END;

END;

FUNCTION Sigmoid( Z: REAL ): REAL;

(\* Сигмоидальная переходная функция нейрона \*)

BEGIN

Sigmoid := 1.0/(1.0+EXP(-Z));

END;

(\* Основная программа \*)

BEGIN

WRITELN('<< P E R C E P T R O N >> (Нейроимитатор) ');

WRITELN('----------------------------------------- ');

VOK := TRUE;

(\* Инициализация с контролем ошибки \*)

RANDOMIZE;

InitAll;

IF (NOT VOK) THEN

BEGIN

WRITELN('Ошибка инициализации');

HALT;

END;

(\* Генерация базы данных \*)

VOK := TRUE;

GetDataBase;

IF (NOT VOK) THEN

BEGIN

WRITELN('Ошибка при генерации базы данных');

HALT;

END;

(\* Цикл обучения \*)

VOK := TRUE;

VCounter := 0;

WITH VNet, VBase DO

REPEAT

VError := 0.0;

(\* Цикл по обучающей выборке \*)

FOR Vi := 1 TO NImages DO

BEGIN

(\* Подача очередного образа на входы сети \*)

FOR Vj := 1 TO NInp DO

BEGIN

Inp[Vj] := Images[Vi].X[Vj];

END;

(\* Цикл по нейронам. При аппаратной реализации

будет выполняться параллельно !!! \*)

FOR Vk := 1 TO NOut DO

BEGIN

(\* Состояние очередного нейрона \*)

VTemp := CBiasNeuron\*W[0,Vk];

FOR Vj := 1 TO NInp DO

BEGIN

VTemp := VTemp +

Inp[Vj]\*W[Vj,Vk];

END;

Out[Vk] := Sigmoid( VTemp );

(\* Накопление ошибки \*)

VDelta := Images[Vi].Y[Vk]-Out[Vk];

VError := VError + 0.5\*SQR( VDelta );

(\* Обучение по дельта-правилу Розенблатта \*)

W[0,Vk] := W[0,Vk] +

CEta\*CBiasNeuron\*VDelta;

FOR Vj := 1 TO NInp DO

BEGIN

W[Vj,Vk] := W[Vj,Vk] +

CEta\*Inp[Vj]\*VDelta;

END;

END;

END;

VCounter := VCounter + 1;

UNTIL ( (VCounter >= CCounter) OR

(VError <= CError) );

(\* Цикл завершен при достижении максимального числа

итераций или минимально достаточной ошибки \*)

WRITELN( 'Выполнено ', VCounter, ' итераций');

WRITELN( 'Ошибка обучения ', VError );

(\* Сохранение результатов обучения на диске \*)

SaveNet;

IF (NOT VOK) THEN

BEGIN

WRITELN('Ошибка при записи на диск');

HALT;

END;

WRITE('Нейронная сеть обучена, параметры');

WRITELN(' записаны в файл SAMPLE.DAT');

END.

**Результат работы программы PERC**

<< P E R C E P T R O N >> (Нейроимитатор)

-----------------------------------------

Выполнено 243 итераций

Ошибка обучения 4.9997994218E-03

Нейронная сеть обучена, параметры записаны в файл

SAMPLE.DAT

**Текст программы TEST.**

PROGRAM TEST;

(\* T E S T - Тестирующая программа для

нейроимитатора PERC \*)

CONST

CMaxInp = 20;

CMaxOut = 10;

CMaxImages = 15;

CBiasNeuron = 1.0;

TYPE

TMatrix = ARRAY[0..CMaxInp,1..CMaxOut] OF REAL;

TInpVector = ARRAY[1..CMaxInp] OF REAL;

TOutVector = ARRAY[1..CMaxOut] OF REAL;

TPerceptron = RECORD

NInp : INTEGER;

NOut : INTEGER;

Inp : TInpVector;

Out : TOutVector;

W : TMatrix;

END;

VAR

VNet : TPerceptron;

VTemp : REAL;

VCorrect : REAL;

Vi, Vj, Vk : INTEGER;

VOK : BOOLEAN;

VFile : FILE OF TPerceptron;

PROCEDURE LoadNet;

(\* Чтение параметров нейронной сети из файла SAMPLE.DAT.

Производится контроль за операциями ввода с использованием

ключа I+ и I- компилятора ТУРБО ПАСКАЛЯ \*)

BEGIN

ASSIGN( VFile, 'SAMPLE.DAT' );

{$I-}

RESET( VFile );

{$I+}

VOK := (IOResult = 0);

IF VOK THEN

BEGIN

{$I-}

READ( VFile, VNet );

CLOSE ( VFile );

{$I+}

VOK := (IOResult = 0);

END;

END;

FUNCTION Sigmoid( Z: REAL ): REAL;

BEGIN

Sigmoid := 1.0/(1.0+EXP(-Z));

END;

BEGIN

VOK := TRUE;

RANDOMIZE;

(\* Чтение параметров обученной нейросети \*)

LoadNet;

IF (NOT VOK) THEN

BEGIN

WRITELN('Ошибка при чтении файла');

HALT;

END;

VOK := TRUE;

WITH VNet DO

BEGIN

WRITELN('<<P E R C E P T R O N>> (Тестирующая программа)');

WRITELN('-----------------------------------------------');

WRITELN(' ВОПРОС ОТВЕТ ВЕРНЫЙ ОТВЕТ ');

WRITELN('-----------------------------------------------');

FOR Vi := 1 TO CMaxImages DO

BEGIN

(\* Подача на вход случайного образа \*)

Vk := 0;

FOR Vj:=1 TO NInp DO

BEGIN

(\* Случайно 0 или 1 \*)

Inp[Vj] := RANDOM( 2 );

(\* Подсчет единиц \*)

IF ( Inp[Vj] > 0 )

THEN Vk := Vk + 1;

END;

(\* Правильный ответ известен ! \*)

IF ( Vk > (NInp-Vk) )

THEN VCorrect := 1.0

ELSE VCorrect := 0.0;

(\* Ответ выдает нейросеть \*)

FOR Vk := 1 TO NOut DO

BEGIN

VTemp := CBiasNeuron\*W[0,Vk];

FOR Vj := 1 TO NInp DO

BEGIN

VTemp := VTemp +

Inp[Vj]\*W[Vj,Vk];

END;

Out[Vk] := Sigmoid( VTemp );

END;

(\* Выдача результатов \*)

FOR Vj := 1 TO NInp DO

WRITE( Inp[Vj]:2:0 );

WRITELN(' ',Out[1]:4:2,' ', VCorrect:2:0);

END;

END;

WRITELN('-------------------------------------------------');

END.

**Результат работы программы TEST.**

<<P E R C E P T R O N>> (Тестирующая программа)

-----------------------------------------------

ВОПРОС ОТВЕТ ВЕРНЫЙ ОТВЕТ

-----------------------------------------------

0 0 0 0 1 1 1 1 0 0 0.00 0

0 0 1 0 0 0 0 1 0 1 0.00 0

1 1 0 0 0 0 0 1 0 0 0.00 0

1 1 1 1 0 1 0 1 1 1 1.00 1

0 1 1 1 0 1 1 0 0 0 0.01 0

1 0 1 0 1 0 1 1 1 0 0.99 1

1 0 1 1 1 0 0 1 1 0 0.98 1

1 0 1 1 1 1 0 0 1 1 1.00 1

1 1 0 1 1 1 1 0 1 0 1.00 1

1 1 0 1 1 1 0 0 0 1 1.00 1

0 0 0 0 1 1 0 1 0 1 0.00 0

1 0 0 1 0 0 0 0 0 1 0.00 0

1 0 0 1 0 0 0 1 1 0 0.00 0

0 1 0 1 1 1 0 1 0 0 0.02 0

1 1 1 1 1 1 0 1 1 0 1.00 1

--------------------------------------------

### Задачи

1. При помощи программы PERC можно изучить зависимость решения от объема данных обучающей выборки. Это достигается изменением значения переменной **Nimages** в подпрограмме **GetDataBase**. Попробуйте объяснить ухудшение результатов теста при обучении с постепенным уменьшением числа образов.

2. Модифицируйте программы PERC и TEST, изменив тип переходной функции нейрона. Сравните результаты.

3. Проведите исследование зависимости скорости обучения от темпа (значение **CEta**) и начального значения весов (значение **CInitWeight**). Объясните полученные вами результаты.